(19)日本国特許庁（ＪＰ）　(12)公表特許公報（Ａ）　(11)特許出願公表番号
特表2001－514411
（P2001－514411A）
(43)公表日　平成13年９月11日(2001.9.11)

(51)Int.Cl.⁷	識別記号	ＦＩ	テーマコード(参考)
Ｇ０６Ｆ　9/06	５５０	Ｇ０６Ｆ　9/06	５５０Ｚ　５Ｂ０１７
12/14	３１０	12/14	３１０Ｈ　５Ｂ０７６

審査請求　有　　予備審査請求　有　　（全 33 頁）

(21)出願番号　特願2000－508048(P2000－508048)
(86)(22)出願日　平成10年８月25日(1998.8.25)
(85)翻訳文提出日　平成12年２月28日(2000.2.28)
(86)国際出願番号　ＰＣＴ／ＵＳ９８／１７５５３
(87)国際公開番号　ＷＯ９９／１０７９５
(87)国際公開日　平成11年３月４日(1999.3.4)
(31)優先権主張番号　０８／９１９，８４４
(32)優先日　平成９年８月28日(1997.8.28)
(33)優先権主張国　米国（ＵＳ）
(81)指定国　ＥＰ(ＡＴ，ＢＥ，ＣＨ，ＣＹ，ＤＥ，ＤＫ，ＥＳ，ＦＩ，ＦＲ，ＧＢ，ＧＲ，ＩＥ，ＩＴ，ＬＵ，ＭＣ，ＮＬ，ＰＴ，ＳＥ)，ＪＰ

(71)出願人　マイクロソフト　コーポレイション
ＭＩＣＲＯＳＯＦＴ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国　ワシントン州　98052－6399　レッドモンド　ワン　マイクロソフト　ウェイ　（番地なし）
(72)発明者　ボンド　パリー
アメリカ合衆国　ワシントン州　98059　レントン　ノースイースト　トゥエンティファースト　ストリート　4902
(74)代理人　弁理士　中村　稔　（外９名）

最終頁に続く

(54)【発明の名称】　信頼できない実行可能コードに関するセキュリティ向上

(57)【要約】
信頼されない実行可能なコードプログラム（アプレット又はコントロール）は、ネイティブ、直接実行可能なコードで書かれる。実行可能なコードは、メモリの外側へのリファレンスが制限される予め割り当てられたメモリ範囲（サンドボックス）内にロードされ、外側のメモリに対するリファレンスが、実行可能なコードに追加されたチェック（スニフコード）によって厳重に制限される。信頼されないコードにおける在来のアプリケーションプログラムインターフェース（ＡＰＩ）コールは、ホストシステムのセキュリティの侵害を防止しながら、実行コードがホストオペレーティングシステムにアクセスすることができる変換コードモジュール（サンク）で置換される。コードにおける静的リンクは、コールによってサンクモジュールに置換される。実行中にＡＰＩコールが作られるとき、コントロールはサンクに移送し、ＡＰＩコールがオペレーティングシステムで実行されることが許容されるべきか判断する。

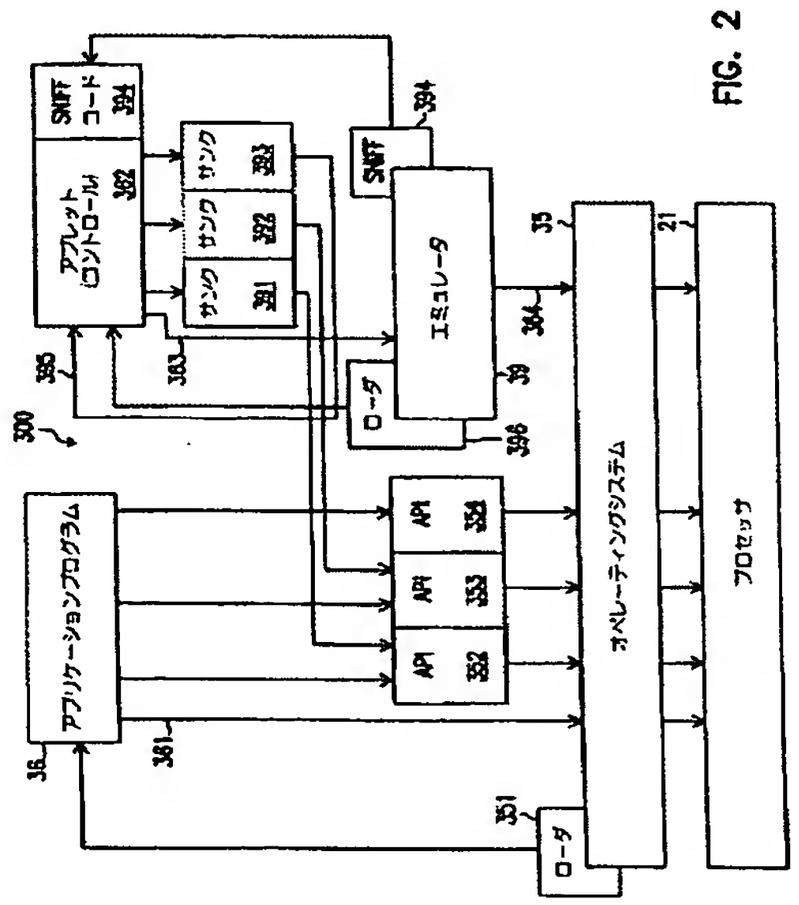

FIG. 2

【特許請求の範囲】

【請求項１】

メモリ及びインターフェースモジュールを有するパーソナルコンピュータプラットフォームで直接実行するために書かれた信頼されないプログラムを実行するための方法であって、

信頼されないプログラムに関してメモリの所定の境界領域を割り当て、

信頼されないプログラムを境界メモリ領域内にロードし、

境界領域の外側のメモリに対してリファレンスをブロックするための信頼されたプログラム内にチェックコードを配置し、

インターフェースモジュールの所定のひとつの実行をパスし、ブロックするための変換コードモジュールに対するリンクを備えるインターフェースモジュールに対するコードにリンクを置換し、

信頼されないプログラムを実行する、

ステップからなる方法。

【請求項２】

境界メモリ領域が更に、信頼されなプログラムに関するワーキングストレージの領域を包含することを特徴とする、請求項１に記載の方法。

【請求項３】

チェックコードが、全体としてメモリアドレスのブロックで作動することを特徴とする、請求項２に記載の方法。

【請求項４】

変換コードモジュールの第１のセットが、インターフェースモジュールの対応するセットに直接アクセスすることを特徴とする、請求項１に記載の方法。

【請求項５】

変換コードモジュールの第２のセットが、インターフェースモジュールの第２の対応するセットにアクセスすることを特徴とする、請求項４に記載の方法。

【請求項６】

信頼されないプログラムをストアするためのメモリの所定の境界領域を割り当て、

信頼されないプログラムを境界メモリ領域内にロードし、

境界領域の外側のメモリに対してリファレンスをブロックするための信頼されたプログラム内にチェックコードを配置し、

インターフェースモジュールの所定のひとつの実行をパスし、ブロックするための変換コードモジュールに対するリンクを備えるインターフェースモジュールに対するコードにリンクを置換する、

コンピュータ実行可能命令コードを包含するコンピュータ読み取り可能記憶媒体。

【請求項７】

ネイティブプロセッサと、

メモリと、

該ネイティブプロセッサと、

該ネイティブプロセッサによって直接実行可能であるネイティブコードで書かれた信頼されないプログラムを実行するためのエミュレータと、

オペレーティングシステムのインターフェースモジュールを使用することによって実行可能であるオペレーティングシステムと、

を有するコンピュータシステムであって、

エミュレータが、

信頼されないプログラムをメモリの境界領域内にロードするためのロードモジュールと、

メモリの境界領域の外で、信頼されないプログラムによるアクセスを制限するための信頼されないプログラム内に挿入可能なチェックコードと、

オペレーティングシステムのインターフェースモジュールの対応するセットにアクセスするための信頼されないプログラムにリンク可能な変換コードモジュールのセットと、

を備える、コンピュータシステム。

【請求項８】

メモリ及びオペレーティングシステムを備えるネイティブコンピュータプラットフォームで直接実行するように書かれた信頼されないプログラムを実行するた

めの方法であって、

信頼されないプログラムに関してメモリの所定の境界領域を割り当て、

信頼されないプログラムを境界メモリ領域内にロードし、

境界領域の外側のメモリに対するリファレンスをブロックするために、信頼されないプログラム内にチェックコードを配置する、

ステップを有する方法。

【請求項９】

信頼されないプログラムに関するランタイムワーキングストレージとして境界メモリ領域の一部を割り当てることを更に有する、請求項８に記載の方法。

【請求項１０】

複数の更なるコードモジュールを境界メモリ領域内にロードすることを更に有し、該コードモジュールが信頼されないプログラムにアクセス可能である、

ことを特徴とする請求項８に記載の方法。

【請求項１１】

メモリ及びオペレーティングシステムを備えるネイティブコンピュータプラットフォームで直接実行するように書かれた信頼されないプログラムを実行するための方法であって、該オペレーティングシステムが、信頼されないプログラムによってリンク可能なインターフェースモジュールのセットを包含し、

インターフェースモジュールの所定のサブセットに対応する変換コードモジュールのセットを構築し、該変換コードモジュールが、サブセットにおけるインターフェースモジュールのそれぞれにコントロールをパスすることができ、

変換コードモジュールの対応するものに対するリンクを備えるインターフェースモジュールに信頼されないプログラムコードのリンクを置換し、信頼されないプログラムによってインターフェースモジュールのあるひとつだけの実行をすることができる、

ステップを有する方法。

【請求項１２】

信頼されないコードにアクセス可能なメモリの境界領域内に変換コードモジュールのセットをストアする、

ことを更に有する請求項１１に記載の方法。

【請求項１３】

アプレットに関するセキュリティを提供する方法であって、

アプレットを予め割り当てられたメモリ範囲内にロードし、予め割り当てられたメモリ範囲が、アプレットをストアするための最初のメモリセグメントと、アプレットによってメモリアクセスが予め割り当てられたメモリ範囲に制限されるようにアプレットの実行中にアクセス可能なストレージに関するランタイムメモリセグメントとの両方を包含し、

危険なＡＰＩに対する各静的なコントロールリンクをサンクＤＬＬと置換し、アプレットによって作られた危険なＡＰＩコールが制限される、

ステップを有する方法。

【請求項１４】

アプレットに関するセキュリティを提供する方法であって、

アプレットを予め割り当てられたメモリ範囲にロードし、

アプレットにおける各静的コントロールリンクをサンクＤＬＬと置換し、

アプレットを実行し、

ＡＰＩコールがアプレットによって作られるとき、コントロールをＤＬＬに転送し、

ＡＰＩコールがオペレーティングシステムで実行することができ、それによってアプレットに関するセキュリティが提供され得るかどうか判断するために、定のセキュリティルールを適用する、

ステップを有する方法。

【請求項１５】

ロードステップの予め割り当てられたメモリ範囲が、コントロールランタイムに関するメモリを包含する、請求項１４に記載のアプレットに関するセキュリティを提供する方法。

【請求項１６】

実行ステップが、予め割り当てられたメモリ範囲に対するコントロールによるメモリアクセスを制限するようにスニフコードを利用する、請求項１４に記載の

アプレットに関するセキュリティを提供する方法。

【請求項１７】

スニフコードが、パフォーマンスを向上するようにメモリブロックで稼働する、請求項１６に記載のアプレットに関するセキュリティを提供する方法。

【請求項１８】

安全なＡＰＩを利用する予め割り当てられたメモリ範囲の外側で、コールを移送するステップを更に包含する、請求項１４に記載のアプレットに関するセキュリティを提供する方法。

【請求項１９】

ウェブベースのアプリケーションを実行するコンピュータシステムに関するセキュリティを提供する方法であって、

ウェブベースのアプリケーションを介して実行可能なコードをダウンロードし、

メモリの所定の領域内に実行可能なコードをロードするためにWx86VMを使用し、

コンピュータシステムのセキュリティが破壊されないように、メモリの所定の領域に対する実行可能なコードの直接のアクセスを晴天するためにWx86VMを使用して、

実行可能なコードのソースが信頼されないソースであるかどうか、

生成された実行可能なコードからソースを判断する、

ステップを有する方法。

【請求項２０】

アプレットを予め割り当てられたメモリ領域内にロードし、該予め割り当てられた領域が、最初のメモリセグメント割り当てと、アプレットによるメモリアクセスが予め割り当てられたメモリ領域に制限されるようなランタイムメモリセグメント割り当てと、の両方を包含し、

アプレットにおける各静的コントロールリンクをサンクＤＬＬで置換し、危険なAPIコールが制限される、

ことを有するステップを実行するためにWx86VMを利用するコンピュータ実行可能

な命令を有するコンピュータ読み取り可能媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、電子的データ処理に関し、特に、信頼できないコードを包含する実行可能プログラムからのシステム損傷を回避することに関する。

【０００２】

【従来の技術】

インターネットブラウザの進歩は、WWW（ワールドワイドウェブ：World Wide Web）の動的及びインタラクティブなページを作り出す。しかしながら、進歩はまた、ウェブページを単に見ることから発生する多くのコンピュータシステムセキュリティリスクを作り出す。インターネットブラウザは、プログラム又は、ウェブページに埋め込まれた他の実行可能なコードを自動的にダウンロードし、実行する。リモートコンピュータからプログラムをダウンロードし、実行する能力は、ホストコンピュータを種々のセキュリティリスクに曝す。例えばコンピュータシステム又は、コンピュータシステムのデータを修正する敵意のあるプログラムは、パスワード、銀行預金口座情報のようなユーザデータを盗み、ユーザにシステムリソースを利用できなくさせる。その結果、セキュリティの問題は、インターネットアプリケーションの開発において重要である。

【０００３】

ある従来技術のアプローチにより、Ｊａｖａアプレットとして知られる実行可能なコードの特定のフォームにセキュリティを設けた。実行可能なコードソースプログラムは、書き込まれ、プラットフォーム独立バイトコードに変換されダウンロードされる。プラットフォーム独立トークン化されたバイトコードは、実行可能コードがすることができる厳格な制限を配置する仮想マシーンで走る。従来技術のアプローチにおける実行可能なコードは、オペレーティングシステムへのアクセスが非常に制限されていた。従って、Ｊａｖａ言語はより強力になるので、オペレーティングシステムがすでに実行できる多くの関数を複製しなければならない。

【０００４】

ActiveXコントロールは、Javaの制限された能力を回避する実行可能なコードのフォームである。ActiveXは、ＯＬＥ（Object Linking and Embedding）及びＣＯＭ（Component Object Model）と呼ばれるマイクロソフト社の２つの技術の産物である。ActiveXは、それがインターネットを利用することができるという特徴をサポートする。例えば、ActiveXコントロールは、Webブラウザによって自動的にダウンロードされ、実行される。

【０００５】

ActiveXコントロールが、ネイティブコードで書かれるので、それらはオペレーティングシステムに十分にアクセスでき、コントロールが稼働するメモリを処理することができる。このアクセスは、コントロールが、スタンドアロンアプリケーションに対する拡張のようなきつく制御された環境で稼働するときに、強力である。しかしながら、ActiveXコントロールが、インターネットエクスプローラのようなウェブブラウザのようなアプリケーションによってインターネット上の知らない又は信用できないソースからダウンロードされるとき、オペレーティングシステムへの十分なアクセスは、深刻なセキュリティの問題を生ずる。ActiveXコントロールは、いかなるオペレーティングシステムのサービスにもアクセスするように設計される。敵意のあるActiveXコントロールは、ホストシステムのハードドライブのお情報を検索することができ、ウィルスを注入することができ、又は、ホストシステムを損傷させることができた。オペレーティングシステムに対するActiveXの無制限のアクセスによる問題は、無制限のアクセスが、セキュリティ違反に対するリスクにホストシステムを置くことである。

【０００６】

従って、ホストシステムのセキュリティを妥協することのない、ホストオペレーティングシステムのパワーにアクセスする能力を備えた実行可能なコードのフォームの必要性がある。

【０００７】

【発明が解決しようとする課題】

本発明は、ネイティブ、即ち直接実行可能なコードで書かれた信頼されたい実行可能なコードに関するセキュリティポリシーを実行する。実行可能なコードは

、メモリの外側へのリファレンスが制限される予め割り当てられたメモリ範囲、即ちサンドボックス内にロードされる。実行中、実行可能なコードに追加されたチェック（「スニフコード（sniff code）」）は、これらの制限を強制する。信頼されないコードにおける在来のアプリケーションプログラムインターフェース（ＡＰＩ）コールは、ホストシステムのセキュリティの侵害を防止しながら、実行コードがホストオペレーティングシステムにアクセスすることができる変換コードモジュール（「サンク（thunks）」）で置換される。コントロール又はアプレットにおける静的リンクは、コールによってサンクモジュールに置換される。実行中にＡＰＩコールが作られるとき、コントロールはサンクに移送し、ＡＰＩコールがオペレーティングシステムで実行されることが許容されるべきか否か判断する。

【０００８】

【課題を解決するための手段】

【発明の実施の形態】

以下の実施の形態の詳細な説明において、添付の図面を参照するが、それらは、本発明を実施するための特定の実施形態の例示として示したものである。これらの実施形態は、当業者が本発明を実施するのに十分に詳細に記載されており、他の実施形態が利用可能であり、本発明の精神及び範囲を逸脱することなく論理的及び電気的な変更をすることができることを理解すべきである。それゆえ、以下の詳細な説明を限定的な意味にとってはならず、本発明の範囲は特許請求の範囲のみによって定義される。複数の図で表される同一のコンポーネントは同じ参照番号によって識別される。

【０００９】

図１及び以下の議論は、本発明を実行することができる適当な計算環境の一般的な説明を短く提供するものである。望まないけれども、本発明は、パーソナルコンピュータによって実行されるプログラムモジュールのようなコンピュータ実行可能な命令の一般的なコンテキストで記載される。一般的に、プログラムモジュールは、特定のタスクを実行し、又は、特定の抽象データ型を実行する、ルーチン、プログラム、オブジェクト、コンポーネント、データ構造などを包含する

。更に、本発明が、ハンドヘルドデバイス、マルチプロセッサシステム、マイクロプロセッサベース又はプログラム可能なカスタマエレクトロニック、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、などを包含する他のコンピュータシステムで実行されうることは、当業者には明らかであろう。本発明はまた、通信ネットワークを介してリンクされたリモート処理デバイスによってタスクが実行される分散計算環境でも実行される。分散計算環境では、プログラムモジュールは、ローカルとリモートの両方のメモリ記憶装置に配置される。

。

【００１０】

図１は、本発明が実施される適当な計算環境の簡単な一般的な説明を提供する。本発明は、以下において、他の環境でも可能であるが、パーソナルコンピュータ（ＰＣ）によって実行されるプログラムモジュールのようなコンピュータ実行可能な命令の一般的なコンテキストとして記載する。プログラムモジュールは、特定のタスクを実行し、特定の抽象データ型を実行するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などを包含する。本発明が、ハンドヘルドデバイス、マルチプロセッサシステム、マイクロプロセッサベース又はプログラム可能なカスタマエレクトロニック、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、などを包含する他のコンピュータシステムで実行されうることは、当業者には明らかであろう。本発明はまた、通信ネットワークを介してリンクされたリモート処理デバイスによってタスクが実行される分散計算環境でも実行される。分散計算環境では、プログラムモジュールは、ローカルとリモートの両方のメモリ記憶装置に配置される。

【００１１】

図１は、本発明を実行するためのシステムの例を示す。それは従来のパーソナルコンピュータ２０のフォームにおいて、汎用のコンピュータデバイスを採用し、該コンピュータは、演算ユニット２１と、システムメモリ２２と、システムメモリ及び他のシステムコンポーネントを演算ユニット２１に接続するシステムバス２３とを含む。システムバス２３は、メモリバス又はメモリコントローラ、周辺バス、及び、ローカルバスを包含する種々のタイプのものであってよく、多数

のバス構造を使用するものであってよい。システムメモリ２２は、ＲＯＭ２４とＲＡＭ２５を包含する。ＲＯＭ２４にストアされた基本入力／出力システム（ＢＩＯＳ）は、パーソナルコンピュータ２０のコンポーネントの間に情報を転送する基本ルーチンを包含する。ＢＩＯＳ２４はまた、システムのスタートアップルーチンを包含する。パーソナルコンピュータは更に、ハードディスク（図示せず）から読み出し、該ディスクに書き込むハードディスクドライブ２７と、リムーバブル磁気ディスク２９から読み出し、該ディスク２９に書き込む磁気ディスクドライブ２８と、ＣＤ－ＲＯＭ又は他の光学媒体のようなリムーバブル光ディスク３１から読み出し、該ディスク３１に書き込む光ディスクドライブ３０とを包含する。ハードディスクドライブ２７、磁気ディスクドライブ２８、及び、光ディスクドライブ３０は、ハードディスクドライブインターフェース３２と磁気ディスクドライブインターフェース３３と、光ディスクドライブインターフェース３４のそれぞれによってシステムバス２３に接続される。ドライブ及びそれらの関係するコンピュータ読み取り可能媒体は、コンピュータ読み取り可能命令、データ構造、プログラムモジュール、及び、パーソナルコンピュータ２０に関する他のデータの不揮発的なストレージを提供する。ここで記載した例示的な環境は、ハードディスク、リムーバブル磁気ディスク２９及びリムーバブル光ディスク３１を採用するけれども、コンピュータによってアクセス可能なデータをストアすることができる他のタイプのコンピュータ読み取り可能媒体をまた具体的な操作環境に使用することもできることは当業者にとって明らかであろう。かかる媒体は、磁気カセット、フラッシュメモリカード、ディジタル汎用ディスク、ベルヌーイカートリッジ、ＲＡＭ、ＲＯＭ及び同様なものを包含する。

【００１２】

プログラムモジュールは、ハードディスク、磁気ディスク２９，光ディスク３１、ＲＯＭ２４、及びＲＡＭ２５にストアされうる。プログラムモジュールは、オペレーティングシステム３５と、１又はそれ以上のアプリケーションプログラム３６と、他のプログラムモジュール３７と、プログラムデータ３８とを包含しうる。ユーザは、ユーザは、キーボード４０及びポインティングデバイス４２のような入力デバイスを介してコマンド及び情報をパーソナルコンピュータに入力

する。他の入力デバイス（図示せず）は、マイクロホン、ジョイスティック、ゲームパッド、衛生アンテナ（サテライト・ディッシュ）、スキャナ等を包含する。これら及び他の入力デバイスはしばしば、システムバス２３に接続されたシリアルポートインターフェース４６を介して演算ユニット２１に接続されるが、それらは、パラレルポート、ゲームポート、又はユニバーサルシリアルバス（ＵＳＢ）のような図１に示されていない他のインターフェースを介して接続されうる。モニタ４７又は他の表示デバイスまたは、ビデオアダプタ４８のようなインターフェースを介してシステムバス２３に接続されうる。モニタに加え、パーソナルコンピュータは典型的には、スピーカ及びプリンタのような楽しゅう偏出力デバイス（図示せず）を含む。

【００１３】

パーソナルコンピュータ２０は、リモートコンピュータ４９のような１又はそれ以上のリモートコンピュータに対して論理的な接続を使用してネットワークされた環境で使用されうる。リモートコンピュータ４９は、他のパーソナルコンピュータ、サーバ、ルータ、ネットワークコンピュータ、ピア・デバイス、又は他の一般的なネットワークノードであってよい。それは典型的には、パーソナルコンピュータ２０と接続する上述の多くの又は全てのコンポーネントを包含するが、記憶装置５０だけを図１に例示した。図１に示した論理的な接続は、ローカルエリアネットワーク（ＬＡＮ）５１及びワイドエリアネットワーク（ＷＡＮ）５２を含む。かかるネットワーク環境は、オフィス、企業の広汎なネットワーク、イントラネット及びインターネットでありふれている。

【００１４】

ＬＡＮネットワーク環境に配置されたとき、ＰＣ２０は、ネットワークインターフェイス又はアダプタ５３を介してローカルネットワーク５１に接続する。インターネットのようなＷＡＮネットワーク環境で使用されるとき、ＰＣ２０は典型的にはネットワーク５２にわたって通信を確立するためのモデム５４又は他の手段を包含する。モデム５４は、ＰＣ２０の内部又は外部にあってよく、シリアルポートインターフェース４６を介してシステムバス２３に接続する。ネットワーク化された環境では、２０内に存在するように示されたプログラムモジュール

又はその一部は、リモート記憶装置５０にストアされうる。もちろん、ネットワーク接続は例示的に示されたものであり、コンピュータ間の通信リンクを確立する他の手段で置換することができる。

【００１５】

本発明では、アプリケーションプログラム３６としてパーソナルコンピュータ２０で稼働する在来のウェブブラウザが、リモートコンピュータ４９からアプレットを自動的にダウンロードする。「アプレット」は短いプログラムであり、通常は単一の関数で実行され、他のアプリケーション内で実行されるように設計されている。アプレットは、それらが必要なときにリモートコンピュータからしばしばダウンロードされ、それらがプライマリアプリケーションによって実行された後、ローカルコンピュータからときどき消去されうる。

【００１６】

図２は、本発明における、アプレットを稼働するためのファシリティを含む殆どが在来の実行環境を示す。用語「アプレット」は、従来技術において正確に定義されていない。この用語は一般的には、単一の関数又は制限された範囲の関数を実行するための小さなプログラムと呼ばれるが、用語は本来は、プログラムのサイズ又はその関数の範囲を制限されない。アプレットは、特定の目的でWWWページのようなオンラインソースからしばしばダウンロードされ、実際には、アプレットはダウンロードされるとすぐに実行され、ついで実行後削除される。以下に記載する好ましい実施形態では、用語「コントロール」又は「ActiveXコントロール」は、アプレットと同意語と考えて良い。ある場合では、発明それ自身が小さなプログラム、ダウンロードされたプログラム、又は、他のいかなるプログラムの特定のフォームで使用することを制限しない。本発明は、「信頼される」ことがないいかなるプログラムについても有用である、即ち、該プログラムとは、システムリソースに十分にアクセスしたならば、システムを損傷させるかも知れない、不確実な出所又は効果のプログラムである。

【００１７】

Windows95のようなオペレーティングシステム３５は、通常のアプリケーションプログラム３６をメモリ内にロードするためのローダモジュール３５１を採用

する。プログラム３６は、ライン３６１によって表されるような演算ユニット２１に命令を直接送信することによって、オペレーティングシステム３５の制御下で実行する。プログラム３６は、アプリケーションプログラムインターフェース（ＡＰＩ）コード３５２－３５４のブロックを呼び出すことによって標準のＡＰＩファンクションを実行する。各ＡＰＩは、図１のディスプレィ４７にダイアログボックスを表示するように、特定のレベルのファンクションを実行するためのプロセッサ２１によって直接実行可能な命令を包含する。ＯＳ３５は、一般的に、数千の独立したＡＰＩを含んでおり、数ダースのダイナミックリンクライブラリ（ＤＬＬ）として通常パッケージングされており、Microsoft WindowsNTオペレーティングシステムでは、これらのDLLは、集合名詞的に「Win32」として知られている。

【００１８】

エミュレータプログラムによって、ある演算ユニット２１用に書かれたアプリケーションプログラムが、異なる命令セットを有する別の演算ユニットで実行される。ここで採用する特定のWx86VMエミュレータ３９は、インテル「ｘ８６」プロセッサ（８０３８６，８０４８６，ペンティアムなど）用に書かれたプログラムをDigital Equipment Corp.のAlpha及びIBMのPowerPCのようなプロセッサで実行するためにオリジナルに開発されたものである。それについては、出願中であるシリアル番号08/912,454及び08/904,057により詳細に記載してある。本目的に関して、Wx86VMと呼ばれるいくぶん修正されたバージョンは、殆どの修正されていない命令をx88プロセッサ２１に通すが、記載するようなその他をブロックし、変換する。Wx86VMは、「サンクコード」（又は単に「サンク」）３９１－３９３と呼ばれる変換モジュールによってＡＰＩを実行する際にWx86をまねるが、ここでのサンクコードの目的は、セキュリティを提供することであり、異なるプラットフォーム用に書かれたＡＰＩコードを実行するために、あるプラットフォームからAPIコールをすることができるというそれらの本来の目的ではない。

【００１９】

３６２のようなアプレットが実行されるとき、インターネットウェブブラウザのようなホストプログラム３６は、エミュレータ３９を呼び出す。エミュレータ

は、アプレットコードを所定のメモリ領域にロードするため、及びそれの使用のために別の所定のメモリ領域を割り当てるために、それ自身のローダモジュール３９６を採用する。これらの領域は、そのアプレットに関する「サンドボックス（sandbox）」と呼ばれる。アプレットの実行中、エミュレータ３９は、アプレットのコードを、サンドボックスの外側に存在するコンパイルされたキャッシュにコンパイルする。コンパイルプロセス中、エミュレータはまた、メモリスニフ（sniff）コード３９４をキャッシュ内に挿入する。

【００２０】

アプレット３６２は、それが書かれた同じプロセッサプラットフォーム２１で実行するので、エミュレータ３９は、ActiveXコントロールを実行するために（ライン３６３で表された）個々の命令を変換する必要がない。しかしながら、それは、セキュリティを提供する目的のためにそれらをフィルタリングし、変換する。例えば、ＡＰＩはオペレーティングシステム３５のカーネルを呼び出すために、ｘ８６割り込み（ＩＮＴ）命令を使用する。それ故、コントロールにおけるINT命令は、ＡＰＩサンク３９１－３９３及びスニフコード３９４をバイパスすることができ、カーネルを直接呼び出すことができる。それ故、エミュレータ３９は、この命令を無条件にブロックし、それは、ライン３６４に全く出力コードを生成しない。ライン３６３でのサブルーチンコール（CALL）及びリターン（RET）、無条件／条件付のジャンプ（JMP/Jxx）のような他の問題のある命令は、サブルーチンコールによってライン３６４に置換され、これらの命令のひとつがシミュレーションされたとき、既にコンパイルされたコードのキャッシュは、コール又はジャンプのキャッシュ内の宛先アドレスを判断するために検索される必要がある。

【００２１】

アプレット３６２からのＡＰＩコールは、APIコード３５２－３５４に直接処理されない。むしろ、サンクコード３９１－３９３はそれらをインターセプトし、それらで何をするか決定する。３９１でのようないくつかのコールは、サンク３９１によって対応するAPI３５２に直接通され、これらのコールは、システムに大混乱をもたらすことはなく、従って、セキュリティリスクが存在しない。３

９２のような他のサンクは、その特定のコールの所定の特徴に依存して、それに対応するAPI353にコールを通すかどうか決定し、それをAPIに出す前にコールを修正することができる。３９３のようなあるサンクは、コールをそれらのAPI354に完全に認めず、これらのコールは、システムのセキュリティを犯し、信頼できないアプレット３６２によって許容されない。

【００２２】

図３は、パーソナルコンピュータ２０で実行されるアプレットがパーソナルコンピュータのセキュリティを有しない全てのオペレーティングシステムサービスにアクセスすることができる発明のある実施形態の大まかなステップ４００を図示する。

【００２３】

ステップ４１０では、ウェブブラウザのようなホストアプリケーションがアプレットを割り当てられたメモリ範囲にロードする。割り当てられたメモリ範囲を、このアプリケーションではサンドボックスと呼ぶ。サンドボックスは、アプレットをストアするための最初のメモリセグメントと、アプレットを実行する間、ストレジをアドレス可能にするためのランタイムメモリセグメントとの両方を含み、これらは在来のいかなる手段でも割り当てられ得る。この実施形態では、OS35は、ステップ４１１でエミュレータ３９を呼び出す。ステップ４１２は、アプレット３６２のコードをストアするために、図１のＲＡＭ２２におけるアドレスの領域及び範囲を割り当て、ランタイムワーキングストレージを使用するためのアプレットに関する他の領域を割り当て、これらの２つの領域は、他のいかなるアプレット、アプリケーションプログラム、又は他のシステムのファシリティに影響を与えることなく、安全に実行することができるサンドボックスを一緒に構成する。それらは、各セキュリティドメインのためのひとつのXW86サンドボックスとなる、即ち同じセキュリティ設定を有する全てのコントロールが同じサンドボックスでプレイする。セキュリティ設定がウェブページのＵＲＬ（uniform resource locator）を含むので、各オープンウェブページは、少なくとも１つのサンドボックスを有する。通常、同じウェブページの全てのコントローラは、同じサンドボックスにある。それらのカスタムインターフェースが安全でないけれど

も、サンドボックス内でインターアプレットを実行することは許容される。

【００２４】

ステップ４２０は、実行のためにアプレットを準備する。

【００２５】

ステップ４２１は、リンクを備えるアプレットの静的リンクをサンクモジュールと置換する。即ち、エミュレータ３９は、アプレット３６２のコード内でAPI352-354に対する全てのコールを見つけ、それらを対応するサンク３９１－３９３に対するコールに変更する。静的リンクは、アプレットの実行中、一定を維持するリンクである。DLL即ちダイナミックリンクライブラリは、実行可能な関数のライブラリ、又は、Windowsアプリケーションによって使用することができるデータである。典型的には、DLLは、１又はそれ以上の特定の関数を提供し、DLLは、DLLに対する静的又は動的なリンクのいずれかを生成することによってアクセスされる。DLLは、最後に拡張子.dllを備える記述でファイルされる。サンクDLLは、サンドボックス内の安全なAPIである。サンクDLLは、安全であると考えられない多くのAPIをブロックし、制限する。例えば、CreateFileが知られたロケーションにだけ許容されうる。同様に、アプレットは、パスワードを記録するための他の処理を生成することができない。上述のように、いくつかのサンクは、対応するAPIにコントロールを単に通す。例えば、「CreateWindow」、「CreateDialog」、「CreateIcon」、「CreateCursor」と名付けられたWin32API及び同様な関数は、他のプロセスに影響せず、信頼できないコードを許容しうる。一方、所定の他のAPIは、信頼できないコードを完全に利用できなくさせなければならない。例えば、「CreateProcess」を許容することにより、信頼できないアプレットをサンドボックスの外側で別のプログラムを実行することができ、「ExitWindowsEx()」のようなオペレーションを完全にブロックすることができ、それにより、信頼できないコードは現在のユーザをログオフすることができず、コンピュータをオフにすることができない。３９３のようなサンクが、ライン３９５によって表示されたコントロールにエラーコードバックを戻すことによってAPIをブロックする。

【００２６】

いくつかのAPIはある条件下、又は所定の修正で許容されうる。この場合、３９２のようなサンクは、それが対応するAPI３５３を呼び出すか又はブロックするかのいずれかであった後、内部演算を実行し、修正されたパラメータをAPIに通す。例えば、「SendMessage()」は通常メッセージをウィンドウに送信する。SendMessageサンクにより、ActiveXコントロールがメッセージをそのコントロールによって生成されたウィンドウに送信することができる。しかしながら、サンクは、ウェブブラウザによって、又は他のアプリケーションプログラムによるそれ自身の全てのメッセージをブロックする。このことにより、コントロールが、他のプログラムに属するウィンドウによって実行されるべきであるキーストロークをまねるためにVM_CHARメッセージを送信することによってセキュリティを侵害することを防止する。

【００２７】

他の例は、メモリをどんな場所にも普通に割り当てる「GlobalAlloc」、「HeapCreate」のようなWin32APIを含む。これらのAPIに関するサンクは、対応するAPIの全体のコードを組み込み、サンドボックスメモリ内で完全に実行するようにリコンパイルし、サンドボックスの境界内のみでメモリを割り当てることができる。

【００２８】

次いで、ステップ４２２は、アプレットのコードを図２のエミュレータ３９によって実行されうるオブジェクトコードにコンパイルする。コードが要求されたものになったとき、コンパイルは直ちに又はパート毎に全て進行し、コンパイルされたコードは、サンドボックスの外側に配置された図４のコンパイルされたキャッシュ３５７に配置される。これらの方法におけるコンパイルは、在来のものであり、発明の本質には関係しない。

【００２９】

ステップ４２３は、認められないメモリリファレンスに対する禁止を強化するためにアプレット自身のコードにチェックコードを挿入する。「スニフ（sniff）コード」と呼ばれるこのチェックコードは、アプレットのコードによって全てのメモリの読み書きを調べ、その結果からそれらを許可し、又は許可しない。ア

プレットがサンドボックスの外側のメモリにアクセスするのを防止することにより、アプレットのセキュリティは向上する。予め割り当てられた範囲からだけのアプレットに対して全てのメモリを提供することにより、スニフコードオーバーヘッドを低減させ、その結果、メモリ範囲の効率的なチェックを生じる。更なる最適化技術が、基本のブロックレベルでコードをコンパイルすることによって追加される。例えば、種々のメモリリファレンスが同じレジスタを使用するアプレットによってなされるならば、コンパイラは、各アクセスに関するスニフコードに対する別々のコールを生成するのではなく、一回だけそのレジスタによってアドレス可能な全体の範囲をチェックすることができる。詳細な例を図４と一緒に示す。基本的には、スニフコードによって、割り当てられたサンドボックス内と、システムを損傷しない所定の他のメモリ内とだけで、アプレットがＲＡＭアドレスを参照することができる。（エミュレータ３９がサンドボックスの外側でメモリを参照することができないが、それはサンドボックスにメモリ領域を割り当てるための能力を有する。デバイス独立ビットマップイメージのような目的に関して、余分のスニフコードオーバーヘッドは労力より小さく、さもなければ最初のサンドボックス領域内にイメージをコピーするように要求される。）

ステップ４３０は、アプレットを実行する。ステップ４３１は、命令シーケンスに続く。

【００３０】

現在の命令がAPIに対するコールであるならば、ステップ４２１によって配置されたリンクは、コールがステップ４３２で完全にブロックされ、ステップ４３３で実行され、ステップ４３４で更に処理され、次いで、ブロックされ又は許可されうるかどうか判断する。

【００３１】

現在の命令が、LOAD又はSTOREのようなメモリリファレンス命令であるならば、ステップ４３５によって、命令がそのサンドボックス内のアドレスを参照するならば、ステップ４３６がその命令を実行することができる。もしそうでなければ、ステップ４３７は、リファレンスがさもなければ許容されるかどうか判断する。もしそうならば、ステップ４３５はそれを実行し、さもなければ、ステップ

４３８はアクセスをブロックし、エラーを返す。スニフコードはこれらのステップを実行する。他のX86命令は、ステップ４３６によって直接実行される。各命令の後、コントロールはステップ４３１に戻る。プロセス４００は、ホストアプリケーションがそれを終了するまで、続く。

【００３２】

いくつかのシステムでは、４３３のようなブロックによってAPIの実行が、他のセキュリティ暴露を現す。APIの引き数がサンドボックスにおけるデータに対するポインタであるならば、サンクがAPIに対して示されたメモリのコンテンツ及び、APIに対する実際のコールを検査する時間の間は、短い時間である。マルチスレッドアプレットでは、アプレット内の他の実行スレッドが、APIに対して示されたメモリのコンテンツを変更し、それによってAPIに対して無効にされたデータを転送することができる。かかるアタックを防止するために、ブロック４３３－１は、APIの引数の「ディープコピー（deep copy）」を実行し、ブロック４３３－２は、APIからの戻り値をディープコピーする。更に特別に、ステップ４３３がAPIを実行するとき、ステップ４３３－１は、APIが実際にコールされる前に、サンドボックス内のそれらの位置から、サンドボックスの外の別の位置に、APIに通された全ての引数を最初にコピーする。アプレット自身がこのコピーにアクセスすることができないので、APIは、既に保存されているデータだけを有効にする。ステップ４３３は、次いで、サンドボックスの外に、戻り値を置き、APIコンポーネントを実行した後、ステップ４３３－２は、アプレットの使用のためにサンドボックスの内側に戻り値をコピーする。所望ならば、ディープコピーが、選択的に使用されうる。

【００３３】

図４は、本発明に関するそれらの領域のみを示すシステムＲＡＭ２５のメモリマップである。予め割り当てられた範囲２５１は、サンドボックスを形成する。それは、アプレット３６２と、アプレットの実行中に、アドレス可能なワーキングストレージに関するランタイムメモリセグメント２５２と、変換コードサンク３９１－３９３（ここではサンク３９１としてだけ示す）をストアするためのセグメントと、をストアするための最初のメモリセグメントを包含する。サンドボ

ックス２５１の外側のメモリ２２は、ここでは３５２によって表されるＡＰＩDLLと、カーネル32３５５とを包含する。他のワーキングメモリ領域は、３５６として表される。コンパイルされたキャッシュ３５７はまた、サンドボックス２１５の外側に配置される。セキュリティポリシーが実際に実行されることがここであるので、サンドボックス２１５の外側のWHKRNL32３２５の位置は、特に重要であり、それがサンドボックスの内側にあるならば、ルージュアプレットが、それを修正することによってセキュリティを妥協することができうる。

【００３４】

以下の例は本発明の作動の例示を示す。前に述べたように、この実施例は、ｘ８６Win32アプレットを実行するための前述のWx86VMエミュレータを利用し、Windows95又はWindowsNTオペレーティングシステム下でｘ８６プラットフォームで修正されずにコントロールする。

【００３５】

マイクロソフトインターネットエクスプローラのようなウェブブラウザは、インターネットからハードドライブのc:¥temp¥foo.ocxに、foo.ocxと呼ばれるActiveXコントロール（アプレット）をダウンロードする。拡張子.ocxはActiveXコントロールを示す。

【００３６】

次いで、インターネットエクスプローラは、システムにおいてWx86VMの存在を探す。Wx86VMコンポーネントが利用可能であるならば、インターネットエクスプローラは、それを呼び出し、コントロールに関する全てのセキュリティに関する情報を提供し、ロードされるべきコントロールを要求する。Wx86VMコンポーネントは、インターネットエクスプローラがそれを提供し、Wx86VMにそれを送り出すか、オブジェクトリンカをOLE32に行かせるかどうか判断するセキュリティ情報を調べ、それを取り扱う。

【００３７】

コントロールがWx86VMエミュレータにおいて送り出されるべきならば、Wx86VMはメモリの割り当てられた領域、又は、ActiveXコントロールに関するサンドボックスを生成する。Wx86VMは、ActiveXコントロール（図４では３６２と示す）f

oo.ocxをサンドボックス内にロードする。

【００３８】

Wx86VMは、３９１のようなAPIサンクDLL（安全API）をサンドボックス内にロードする。Wx86VMは、前述の出願（ドケット777.016US1）において十分に説明したようなオペレーティングシステムローダ内でDLLの名前を修正することができる。このことにより、Wx86VMは、規定を呼び出す際に違いを取り扱うために、x86イメージと本来のAPIとの間にサンクコードを挿入することができる。再配置するための名前のリストは、レジストリにストアされる。例えば、カーネル３２（図４では３５５）は、（図４ではサンク３９１と示す）wikrnl32に再配置され、user32.dllはwiuser32.dllに再配置される。APIサンクは次の２つのDLLから構成される：１つは、Wx86VM内で稼働する「wi」という接頭辞が付いたDLLであり、今後は信頼性がなく、他のひとつはWx86VMの外側で稼働し、「wh」という接頭辞が付いたDLLであり、安全ポリシーを実行するために信頼される。

【００３９】

「wi」DLLは、それらが置換するための本来のDLLとして同じエクスポートを有する。これらのエクスポートは、サンドボックスの外でスイッチングをするためにWx86VMに対して応答可能であり、次いで、安全モードにおいて適当なサンクを呼び出し、これは更に、そのAPIに関してセキュリティポリシーを実行する。特定のAPIに関するセキュリティがないならば、サンクは単に本来のAPIを呼び出す。「BOP」と呼ばれるこのコールは、典型的には、モードスイッチが起こるのを必要とするWx86.dllを合図する無効なx86 opcodeである。BOPコマンドは、フォーム「BOP(DLL#,API#)」を有する。Wx86VMが、DLLがサンドボックスのレジスタセット及びスタックにアクセスする、（図４ではwhkrnl32３５２のような）「wh」という接頭辞であるホストサイドサンクDLLにBOPをディスパッチするとき、DLLは、パラメータをサンドボックスのスタックから本来のスタックにコピーし、APIの引数を検査し、コールを作り、サンドボックスのEAXレジスタに戻り値を戻すように移動させることができる。

【００４０】

例えば、ｘ８６アプレット又はコントロールが、kernel32!CreateFileに対す

る静的なリンクを有するならば、Wx86VMはそのリンクを、wikrnl32!CreateFileと解決する。アプレットがCreateFileを呼び出すとき、wikrnl32!CreateFileは、サンドボックスからネイティブにスイッチするBOP命令を実行し、Wx86IDispatchBop()をWx86VMI.dllに呼び出す。Wx86DispatchBop()は、コールをwhkrnl32!whCreateFile()にディスパッチする。その関数は、ネイティブkernel32!CreateFile()を呼び出し、戻り値をシミュレーションされたEAXレジスタにコピーし、戻る。

【００４１】

Wx86VMはまた、エミュレータ３９コードWx86cpu.dllをロードする。アプレットの実行中、プロセッサがBOP命令に出会うとき、エミュレーションは停止する。

【００４２】

アプレットの実行は、必要とされるコードをコンパイルし、コードをコンパイルされたキャッシュに置くことにより始まる。コンパイルされたコードは、メモリ読み書きオペレーションが安全なオペレーションであることを確認するためにそれにおいてスニフチェックを有する。アクセスされるメモリが、所定のサンドボックス領域の外側であるならば、メモリにアクセスする試みのオペレーションは失敗する。例えば、アプレットfoo.ocxが命令MOV EAX, [ESI+4]を包含するならば、コンパイラは、命令が安全であることを確認するためのMOV命令の前にスニフコードを挿入する。以下の命令：

```
MOV EAX, [ESI+4]
```

は、スニフコードが挿入された後に、

```
LEA ECX, [ESI+4]
CALL SNIFFREAD4.ECX
MOV EAX, [ECX]
```

となる。

【００４３】

スニフコードがオーバーヘッドに加わるので、基本的なブロックレベルでコードをコンパイルするとき、追加の最適化技術は適用されうる。例えば、アプレッ

トが同じレジスタを使用する種々のメモリリファレンスを作るならば、コンパイラは、一回だけ全体の範囲をチェックし、個々のスニフコールを生成しない。アプレットfoo.ocxが以下の命令を包含するならば、

```
MOV EAX, [ESI+4]
MOV EDX, [ESI+8]
```

スニフコードは、以下のように挿入される：

```
LEA EAX, [ESI+4]
CALL SNIFFREAD8.ECX
MOV EAX, [ECX]
MOV EDX, [ESI+4]
```

むしろ、より小さな効率的な仕方でスニフコードを挿入する：

```
LEA EAX, [ESI+4]
CALL SNIFFREAD4.ECX
MOV EAX, [ECX]
LEA EAX, [ESI+4]
CALL SNIFFREAD4.ECX
MOV EDX, [ECX]
```

上の記述は例示的なものであり、制限的なものではない。上の記述をみれば、当業者にとって多くの他の実施形態が明らかである。それ故、本発明の範囲は、特許請求の範囲と均等な範囲とあわせて、特許請求の範囲を参照して決定されるべきである。

【図面の簡単な説明】

【図１】

本発明が実施される、例示的な計算環境のシステム図である。

【図２】

本発明を組み込む実行環境のブロック図である。

【図３】

本発明の主なステップを記述するフローチャートである。

【図４】

メモリ内のサンドボックス領域の簡単化したブロック図である。

【図１】

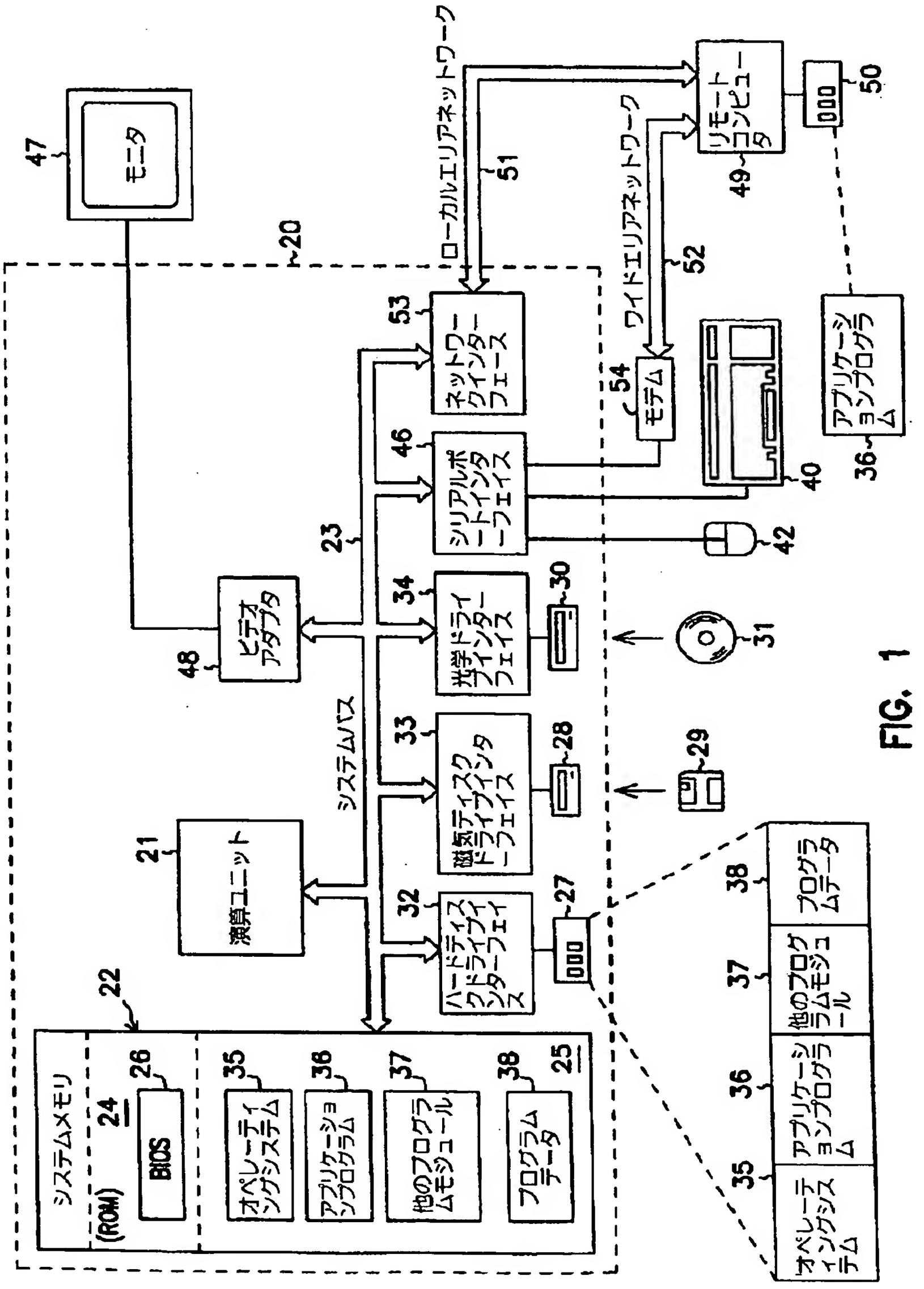

FIG. 1

【図２】

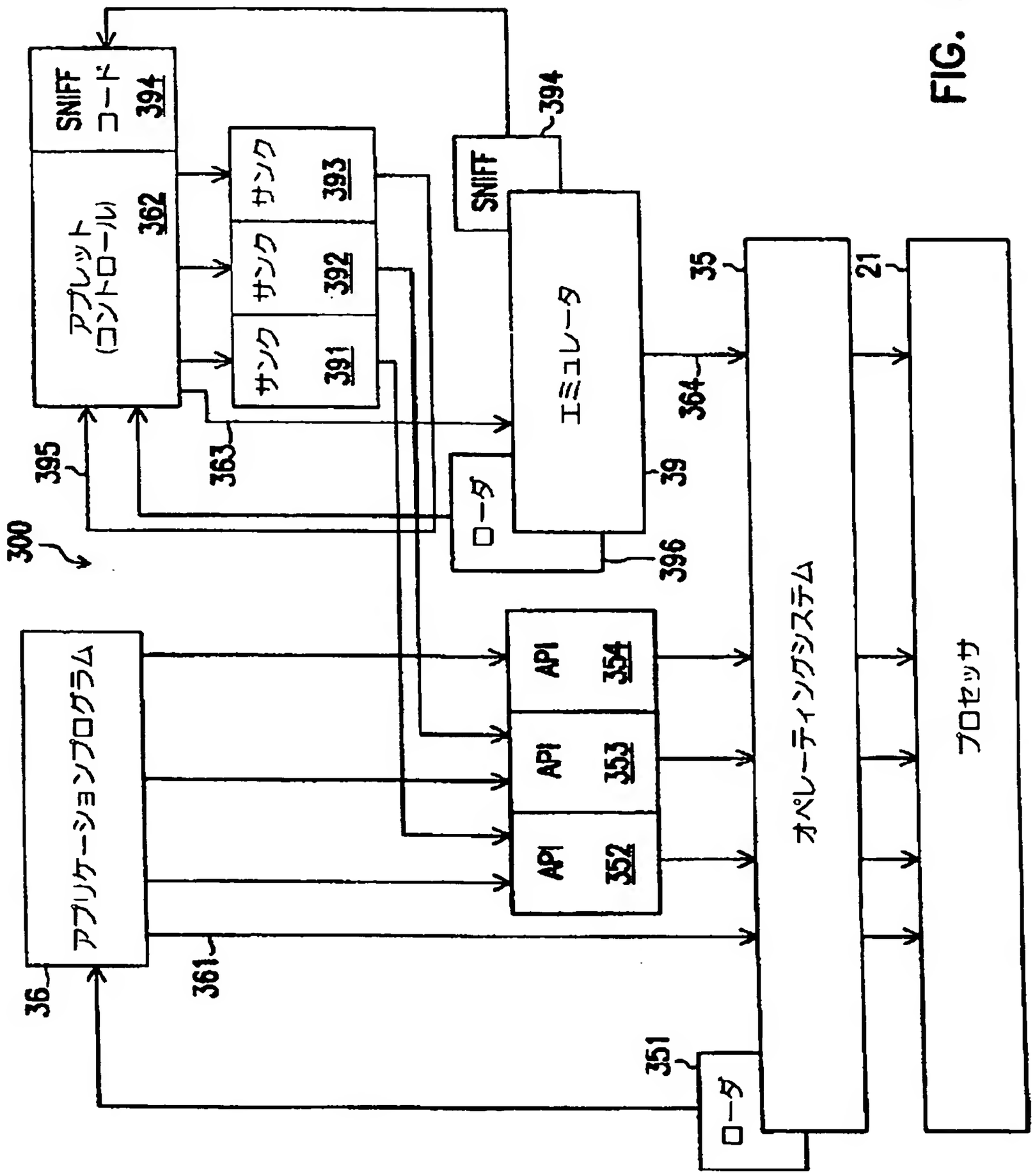
SNIFF
コード
394
アプレット
(コントロール)
362
サンク
393
サンク
392
サンク
391
SNIFF
394
エミュレータ
35
21
364
395
363
ローダ
39
300
396
アプリケーションプログラム
API
354
API
353
API
352
オペレーティングシステム
プロセッサ
36
361
351
ローダ

FIG. 2

【図３】

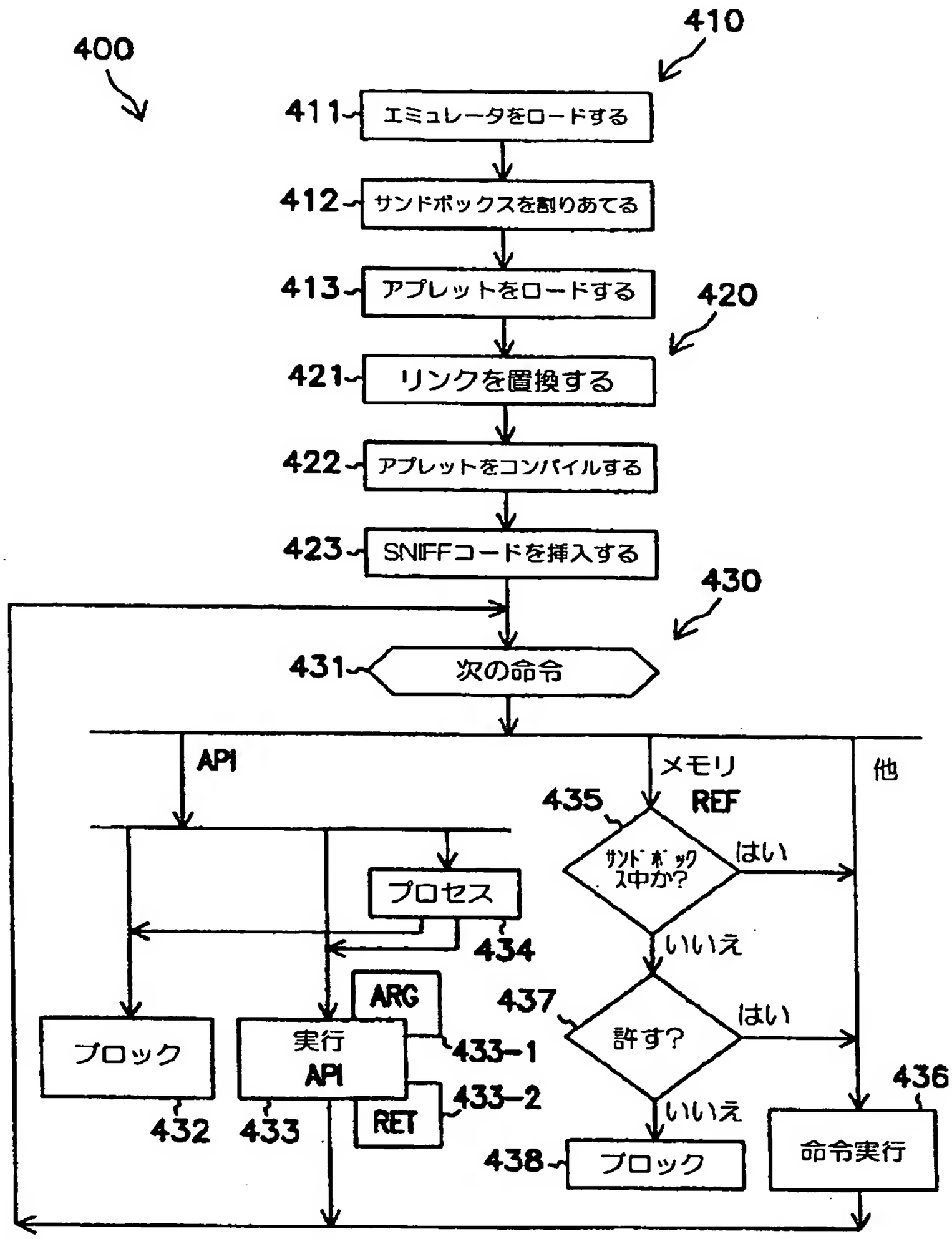
400
410
411
エミュレータをロードする
412
サンドボックスを割りあてる
413
アプレットをロードする
420
421
リンクを置換する
422
アプレットをコンパイルする
423
SNIFFコードを挿入する
430
431
次の命令
API
メモリ
REF
他
435
サンドボックス中か?
はい
プロセス
434
いいえ
ARG
437
許す?
はい
ブロック
実行
API
433-1
433-2
RET
436
いいえ
命令実行
432
433
438
ブロック

FIG. 3

【図４】

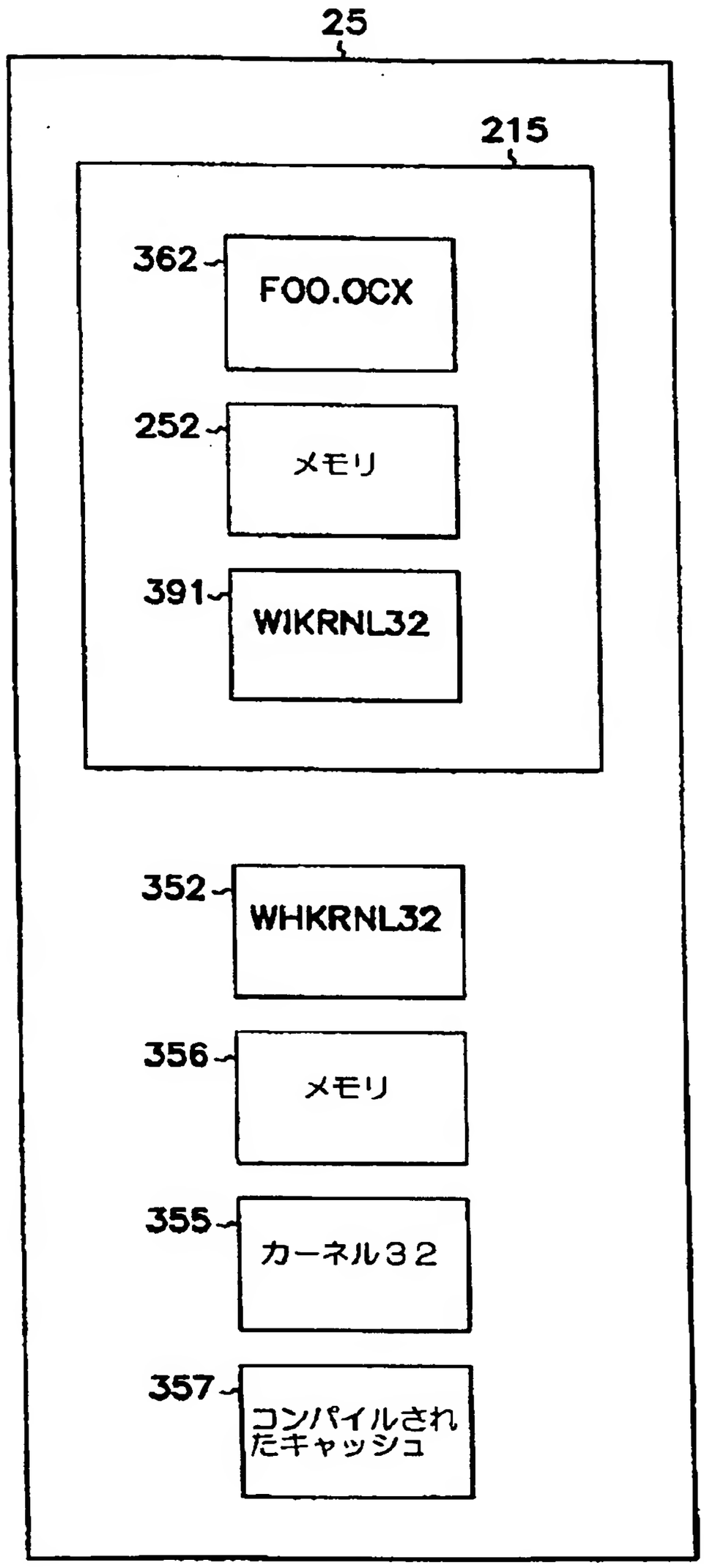
25
215
362
FOO.OCX
252
メモリ
391
WIKRNL32
352
WHKRNL32
356
メモリ
355
カーネル３２
357
コンパイルされたキャッシュ

FIG. 4

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Int. tional Application No
PCT/US 98/17553

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	ESAFE TECHNOLOGIES INC.: "New Anti-Vandal Software Provides 'Next Generation' PC Protection" 28 April 1997 , SAN DIEGO, US, XP002086033 Available from Internet:<URL: http://www.esafe.com/press/pr032997.html> see the whole document ---	1,8,11, 13,15
Y	EP 0 667 572 A (IBM) 16 August 1995 see the whole document ---	1,8,11, 13,15
A	WO 94 07204 A (UNILOC CORP PTY LIMITED ;RICHARDSON RIC BAILIER (AU); UNILOC SINGA) 31 March 1994 see the whole document ---	1,6-8,16

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 November 1998	08/12/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Powell, D

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int. ional Application No
PCT/US 98/17553

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	HAMILTON M A: "JAVA AND THE SHIFT TO NET-CENTRIC COMPUTING" COMPUTER, vol. 29, no. 8, August 1996, pages 31-39, XP000632765 see page 31, paragraph 4 see page 34, left-hand column, paragraph 4 - right-hand column, last paragraph see page 36, left-hand column, paragraph 1 right-hand column, paragraph 1 ---	2,9,14, 15
A	EP 0 646 865 A (BULL HN INFORMATION SYST) 5 April 1995 see abstract; figures 1,25-8 see page 6, line 46 - line 51 see page 21, line 32 - line 46 -----	7

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int. ional Application No
PCT/US 98/17553

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0667572 A	16-08-1995	JP 7230380 A	29-08-1995
		US 5673315 A	30-09-1997
WO 9407204 A	31-03-1994	AU 678985 B	19-06-1997
		AU 4811393 A	12-04-1994
		CA 2145068 A	31-03-1994
		CN 1103186 A	31-05-1995
		EP 0689697 A	03-01-1996
		NZ 255971 A	26-05-1997
		US 5490216 A	06-02-1996
EP 0646865 A	05-04-1995	AU 679775 B	10-07-1997
		AU 7428994 A	13-04-1995
		CA 2132900 A	29-03-1995
		JP 7182180 A	21-07-1995
		US 5572711 A	05-11-1996
		US 5675771 A	07-10-1997
		US 5566326 A	15-10-1996
		US 5664098 A	02-09-1997

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(72)発明者　バラッティー　スディープ
アメリカ合衆国　ワシントン州　98008
ベルヴィュー　ワンハンドレッドアンドシックスティフィフス　プレイス　ノースイースト　3272

Ｆターム(参考)　5B017 AA07 BA06 BB06 CA01
5B076 FD00